Europäisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

99480101.7

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

I.L.C. HATTEN-HECKMAN

DEN HAAG, DEN
THE HAGUE, 22/11/99
LA HAYE, LE

Europäisches Patentamt | European Patent Office | Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.: 99480101.7
Demande n°:

Anmeldetag:
Date of filing: 21/10/99
Date de dépôt:

Anmelder:
Applicant(s):
Demandeur(s):
INTERNATIONAL BUSINESS MACHINES CORPORATION
Armonk, NY 10504
UNITED STATES OF AMERICA

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
System and method for accessing a socks server from an end user workstation in an IP network

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat: State: Pays:	Tag: Date: Date:	Aktenzeichen: File no. Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
/

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE
Etats contractants désignés lors du depôt:

Bemerkungen:
Remarks:
Remarques:

SYSTEM AND METHOD FOR ACCESSING A SOCKS SERVER FROM AN END USER WORKSTATION IN AN IP NETWORK

Technical field of the invention

The present invention relates to computer networks, and more particularly to a method and system for setting the value of the Type Of Service (TOS) field of an IP Datagram according to the Application Level protocol used by Socks data in an end user workstation attached to an Internet Protocol (IP) network.

Background art

INTERNET

The Internet is a global network of computers and computers networks (the "Net"). The Internet connects computers that use a variety of different operating systems or languages, including UNIX, DOS, Windows, Macintosh, and others. To facilitate and allow the communication among these various systems and languages, the Internet uses a language referred to as TCP/IP ("Transmission Control Protocol/Internet Protocol"). TCP/IP protocol supports three basic applications on the Internet :

- transmitting and receiving electronic mail,
- logging into remote computers (the "Telnet"), and
- transferring files and programs from one computer to another ("FTP" or "File Transfer Protocol").

TCP/IP

The TCP/IP protocol suite is named for two of the most important protocols:

- a Transmission Control Protocol (TCP), and
- an Internet Protocol (IP).

Another name for it is the Internet Protocol Suite. The more common term TCP/IP is used to refer to the entire protocol suite. The first design goal of TCP/IP is to build an interconnection of networks that provide universal communication services: an *internetwork*, or *internet*. Each physical network has its own technology dependent communication interface, in the form of a programming interface that provides basic communication functions running between the physical network and the user applications. The architecture of the physical networks is hidden from the user. The second goal of TCP/IP is to interconnect different physical networks to form what appears to the user to be one large network.

TCP is a transport layer protocol providing end to end data transfer. It is responsible for providing a reliable exchange of information between 2 computer systems. Multiple applications can be supported simultaneously over one TCP connection between two computer systems.

IP is an internetwork layer protocol hiding the physical network architecture bellow it. Part of the communicating messages between computers is a routing function that ensures that messages will be correctly directed within the network to be delivered to their destination. IP provides this routing function. An IP message is called an IP Datagram.

Application Level protocols are used on top of TCP/IP to transfer user and application data from one origin computer system to one destination computer system. Such Application Level protocols are for instance File Transfer Protocol (FTP), Telnet, Gopher, Hyper Text Transfer Protocol (HTTP).

IP ROUTER

A "Router" is a computer that interconnects two networks and forwards messages from one network to the other. Routers are able to select the best transmission path between networks. The basic routing function is implemented in the IP layer of the TCP/IP protocol stack, so any host (or computer) or workstation running TCP/IP over more than one interface could, in theory, forward messages between networks. Because IP implements the basic routing functions, the term "IP

Router" is often used. However, dedicated network hardware devices called "Routers" can provide more sophisticated routing functions than the minimum functions implemented in IP.

WORLD WIDE WEB

With the increasing size and complexity of the Internet, tools have been developed to help find information on the network, often called navigators or navigation systems. Navigation systems that have been developed include standards such as Archie, Gopher and WAIS. The World Wide Web ("WWW" or "the Web") is a recent superior navigation system. The Web is :

- an Internet-based navigation system,
- an information distribution and management system for the Internet, and
- a dynamic format for communicating on the Web.

The Web seamlessly, for the use, integrates format of information, including still images, text, audio and video. A user on the Web using a graphical user interface ("GUI", pronounced "gooey") may transparently communicate with different host computers on the system, and different system applications (including FTP and Telnet), and different information formats for files and documents including, for example, text, sound and graphics.

HYPERMEDIA

The Web uses hypertext and hypermedia. Hypertext is a subset of hypermedia and refers to computer-based "documents" in which readers move from one place to another in a document, or to another document, in a non-linear manner. To do this, the Web uses a client-server architecture. The Web servers enable the user to access hypertext and hypermedia information through the Web and the user's computer. (The user's computer is referred to as a client computer of the Web Server computers.) The clients send requests to the Web Servers, which react, search and respond. The Web allows client application software to request and receive hypermedia

documents (including formatted text, audio, video and graphics) with hypertext link capabilities to other hypermedia documents, from a Web file server.

The Web, then, can be viewed as a collection of document files residing on Web host computers that are interconnected by hyperlinks using networking protocols, forming a virtual "web" that spans the Internet.

UNIFORM RESOURCE LOCATORS

A resource of the Internet is unambiguously identified by a Uniform Resource Locator (URL), which is a pointer to a particular resource at a particular location. A URL specifies the protocol used to access a server (e.g. HTTP, FTP,..), the name of the server, and the location of a file on that server.

HYPER TEXT TRANSFER PROTOCOL

Each Web page that appears on client monitors of the Web may appear as a complex document that integrates, for example, text, images, sounds and animation. Each such page may also contain hyperlinks to other Web documents so that a user at a client computer using a mouse may click on icons and may activate hyperlink jumps to a new page (which is a graphical representation of another document file) on the same or a different Web server.

A Web server is a software program on a Web host computer that answers requests from Web clients, typically over the Internet. All Web use a language or protocol to communicate with Web clients which is called Hyper Text Transfer Protocol ("HTTP"). All types of data can be exchanged among Web servers and clients using this protocol, including Hyper Text Markup Language ("HTML"), graphics, sound and video. HTML describes the layout, contents and hyperlinks of the documents and pages. Web clients when browsing :

- convert user specified commands into HTTP GET requests,
- connect to the appropriate Web server to get information, and

- wait for a response. The response from the server can be the requested document or an error message.

After the document or an error message is returned, the connection between the Web client and the Web server is closed.

First version of HTTP is a stateless protocol. That is with HTTP, there is no continuous connection between each client and each server. The Web client using HTTP receives a response as HTML data or other data. This description applies to version 1.0 of HTTP protocol, while the new version 1.1 break this barrier of stateless protocol by keeping the connection between the server and client alive under certain conditions.

BROWSER

After receipt, the Web client formats and presents the data or activates an ancillary application such a sound player to present the data. To do this, the server or the client determines the various types of data received. The Web Client is also referred to as the Web Browser, since it in fact browses documents retrieved from the Web Server.

DOMAIN NAMES

The host or computers names (like www.entreprise.com) are translated into numeric Internet addresses (like 194.56.78.3), and vice versa, by using a method called DNS ("Domain Name Service"). DNS is supported by network-resident servers, also known as domain name servers or DNS servers.

INTRANET

Some companies use the same mechanism as the Web to communicate inside their own corporation. In this case, this mechanism is called an "Intranet". These companies use the same networking/transport protocols and locally based Web servers to provide access to vast amount of corporate information in a cohesive fashion. As this data may be private to the corporation, and because the members of the company

still need to have access to public Web information, to avoid that people not belonging to the company can access to this private Intranet coming from the public Internet, they protect the access to their network by using a special equipment called a Firewall.

FIREWALL

A Firewall protects one or more computers with Internet connections from access by external computers connected to the Internet. A Firewall is a network configuration, usually created by hardware and software, that forms a boundary between networked computers within the Firewall from those outside the Firewall. The computers within the Firewall form a secure sub-network with internal access capabilities and shared resources not available from the outside computers.

Often, the access to both internal and external computers is controlled by a single machine, said machine comprising the Firewall. Since the computer, on which the Firewall is, directly interacts with the Internet, strict security measures against unwanted access from external computers are required.

A Firewall is commonly used to protect information such as electronic mail and data files within a physical building or organization site. A Firewall reduces the risk of intrusion by unauthorized people from the Internet. The same security measures can limit or require special software for people inside the Firewall who wish to access information on the outside. A Firewall can be configured using "Proxies" or "Socks" to control the access to information from each side of the Firewall.

PROXY SERVER

A HTTP Proxy is a special server that typically runs in conjunction with Firewall software and allows an access to the Internet from within a Firewall. The Proxy Server :

- waits for a request (for example a HTTP request) from inside the Firewall,

- forwards the request to the remote server outside the Firewall,
- reads the response, and
- sends the response back to the client.

A single computer can run multiple servers, each server connection identified with a port number. A Proxy Server, like an HTTP Server or a FTP Server, occupies a port. Typically, a connection uses standardized port numbers for each protocol (for example, HTTP = 80 and FTP = 21). That is why an end user has to select a specific port number for each defined Proxy Server. Web Browsers usually let the end user set the host name and port number of the Proxy Servers in a customizable panel. Protocols such as HTTP, FTP, Gopher, WAIS, and Security can usually have designated Proxies. Proxies are generally preferred over Socks for their ability to perform caching, high-level logging, and access control, because they provide a specific connection for each network service protocol.

SOCKS AND SOCKS SERVER

Socks is a protocol which does some form of encapsulation of Application Level protocols (for instance FTP, Telnet, Gopher, HTTP). Using Socks, the Application Level traffic between a system running a Socks Client software and a system running a Socks Server software is encapsulated in a virtual Socks tunnel between both systems. Socks is mainly used by systems within an Intranet in order to gain a secure access to systems located outside the Intranet.

A Socks Server acts as a relay between the systems within the Intranet and the systems outside the Intranet, thus hiding the internal systems from the external Internet. It is considered as one form of Firewall.

A Socks Server (also called Socks Gateway) is a software that allows computers inside a Firewall to gain access to the Internet. A Socks Server is usually installed on a server positioned either inside or on the Firewall. Computers within the Firewall access the Socks Server as Socks Clients to reach the Internet. Web Browsers usually let the end user set the

host name and port number of the Socks Servers in a customizable panel. On some Operating Systems, the Socks Server is specified in a separate file (e.g. socks.conf file). As the Socks Server acts a layer underneath the protocols (HTTP, FTP, ..), it cannot cache data (as Proxy does), because it doesn't decode the protocol to know what kind of data it transfers.

DISPATCHER SYSTEM

When multiple Firewalls are used to gain access to systems outside the Intranet, a dedicated device called "Dispatcher System" is often used within the Intranet for dispatching the traffic to these multiple Firewalls. The main goal of the Dispatcher System is to balance the load across the multiple Firewalls. For instance when a very powerful Firewall and a smaller Firewall are available, more traffic should be dispached on the very powerful Firewall than on the smaller one. Such Dispatcher Systems are either dedicated hardware devices, or software components installed on existing network device (such as an IP Router).

More explanations about the technical field presented in the above sections can be found in the following publications incorporated herewith by reference:

- "TCP/IP Tutorial and Technical Overview" by Martin W. Murhammer, Orcun Atakan, Stefan Bretz, Larry R. Pugh, Kazunari Suzuki, David H. Wood, International Technical Support Organization, October 1998, GG24-3376-05.
- "Java Network Programming" by Elliotte Rusty Harold, published by O'Reilly, February 1997.
- "Internet in a nutshell" by Valerie Quercia, published by O'Reilly, October 1997.
- "Building Internet Firewalls" by Brent Chapman and Elizabeth Zwichky, published by O'Reilly, September 1995.

PROBLEM

The problem is to differentiate the IP Datagrams transporting Socks data according to the Application Level Protocol used by said Socks data. By nature, the Socks protocol is a form of encapsulation of Application Level traffic such as HTTP, FTP, Telnet. When Socks Servers are used within an Intranet to provide secure access to systems located outside the Intranet, IP routers and network devices within this Intranet only see and handle Socks traffic. As a consequence, all Application Level protocols encapsulated by Socks are treated alike within the TCP/IP network.

When multiple Socks Servers are used within the Intranet to access systems outside the Intranet, a dedicated device called "Dispatcher System" is often used for dispatching the traffic on these multiple Socks Servers. The purpose of such Dispatcher System is mainly to balance the load across the multiple Socks Servers. For instance when a very powerful Socks Server and a smaller Socks Servers are available, more traffic can be dispatched on the very powerful Socks Server than on the smaller one. In a Socks environment, the Dispatcher System usually only sees and process Socks traffic and does not see the Application Level traffic which is encapsulated by Socks. As a consequence, all Application Level protocols such as HTTP, FTP, Telnet, are then treated alike by the Dispatcher System. To solve this problem, the Dispatcher System and any other IP network devices such as IP Routers can use the Type Of Service (TOS) field in the Header of the IP Datagrams comprising Socks data. For instance, based on the TOS field, interactive Telnet traffic can be processed by a Dispatcher System with a higher priority than batch FTP traffic. For instance, this Telnet traffic can be dispatched on a high capacity Socks Server while the FTP traffic is dispatched on a smaller Socks Server.

The problem is then to set the value of the Type Of Service (TOS) field in IP Datagrams comprising Socks data, according to the Application Level protocol used in said IP Datagrams.

The current solutions address this problem partially:

- The Type Of Service (TOS) field can be set by any network device according to the protocol of the TCP data transported in the IP Datagrams. That protocol is identified by the Destination Port field of the TCP Header within the IP Datagram. The major drawback is:

  - In a Socks environment, the protocol of the TCP data transported in IP Datagrams is always the Socks protocol. As a consequence, the TOS field has always the same value which corresponds to the Socks protocol. The TOS field is therefore not representative of the Application Level protocol of Socks data. For instance, IP Datagrams transporting HTTP, FTP, or Telnet data over Socks have all the same TOS value. The Type Of Service (TOS) field cannot be used to differentiate the IP Datagrams according to the Application Level protocol of Socks data since this field has always the same unique value.

- The Type Of Service (TOS) field can be set by a network device, for instance a Router. The main drawbacks of this solution are the following:

  - The association between Type Of Service (TOS) value and Application Level protocol is then done by a network device within the Intranet, and not by the end user workstation. Some Service Providers may not use such network device within their IP network. As a consequence, companies which are not owner of their Intranet network but which lease it to such Service Providers, will not take advantage of this solution.

  - The network devices have to maintain connection tables to associate the TOS value with the Application Level protocol of each IP Datagram transporting Socks data. Maintaining such tables requires some processing capacity and therefore has an impact on the performance of the network devices. Furthermore, such tables require a large

amount of memory on these network devices, which is very costly.

- To each Application Level protocol corresponds one and only one TOS value. However, some groups of end users may need to use specific Type Of Service (TOS) values different to the Type Of Service (TOS) values already used by the other end users for the same Application Level protocols. This is not possible with this solution. For example, all end users within the Intranet will use the same low priority (deducted from TOS field within the network) for their FTP traffic, although one group may specifically need a high priority for its FTP traffic.

Objects of the invention

- An object of the present invention is to set the value of the Type Of Service (TOS) field in IP Datagrams comprising Socks data, according to the Application Level protocol used by said Socks data from an end user workstation within an IP network.

- It is a further object of the present invention to differentiate Socks traffic using the Type Of Service (TOS) field.

- It is another object of the present invention to optimize performance and availability of WEB access via Socks Servers, by differentiating Socks traffic based on the TOS field value.

Summary of the invention

The present invention discloses a method and system, in a source device, for setting the value of a Type of Service (TOS) field in an Internet Protocol (IP) datagram according to the application protocol level of socks data transported in

this IP datagram. This IP datagram is sent from a source application on a source device to a destination application on a destination device. The present method comprises the steps of:

- identifying the source device,
- identifying the destination device,
- identifying the application on the source device,
- identifying the application on the destination device,
- determining a Type of Service (TOS) value referring to a first table, said first table comprising for each socks connection identified by:
  - a source device address,
  - a destination device address,
  - a source application address, and
  - a destination application address,

  a Type Of Service (TOS) value.
- writing the Type Of Service (TOS) value in the Type Of Service (TOS) field of the IP datagram.

If the IP datagram comprises a message for establishing a new socks connection, the step of determining a Type Of Service (TOS) value comprises the preliminary steps of:

- updating the first table with the new socks connection identified by:
  - the source device address,
  - the destination device address,
  - the source application address,
  - the destination application address,

  of the IP datagram,
- identifying the application level protocol from the IP datagram,

- determining a Type of Service (TOS) value referring to a second table, said second table comprising for each

application level protocol a Type Of Service (TOS) value,

- associating in said first table said socks connection with said Type Of Service (TOS) value.

The method comprises the further step of configuring this second table, this second table being preferably retrieved from a server system within the network.

Drawings

The novel and inventive features believed characteristics of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative detailed embodiment when read in conjunction with the accompanying drawings, wherein :

- Figure 1 is a logical view of an end user system accessing the World Wide Web, according to prior art.
- Figure 2 is a general view of an end user system accessing the World Wide Web according to prior art.
- Figure 3 shows an IP Datagram according to prior art.
- Figure 4 is a general view of a Socks Client with a plurality of Socks Servers according to prior art.
- Figure 5 is a general view of an end user workstation retrieving from an Auto TOS System the Type Of Service (TOS) values associated with the Application Level protocols used by Socks data according to the present invention.
- Figure 6 is an internal view of an end user workstation according to the present invention.

- Figure 7 shows the tables used by the Socks TOS Client according to the present invention.

- Figure 8 is a flow chart of the TOS Definition Builder component, according to the present invention.

- Figure 9 is a flow chart of the Socks Traffic Analyser according to the present invention.

- Figure 10 is a flow chart of the Socks TOS Finder according to the present invention.

- Figure 11 is a flow chart of the Socks TOS Builder according to the present invention.

Preferred embodiment of the invention

ACCESS TO THE WORLD WIDE WEB

Logical View

Figure 1 shows a user system with a user interface (102) comprising a Web Browser (101) for accessing the World-Wide-Web (WWW). The WWW content is transferred using the HTTP protocol. HTTP requests and responses are exchanged between the Web Browser program (101) and a destination Web Server (103) containing the WWW information the user wants to access. The Socks Server (104) between the Web Browser (101) and the Web Server (103) acts as an intermediary HTTP Relay forwarding the HTTP requests and responses to their destination. The Web Browser program (101) makes an HTTP request to the Socks Server (104) and the Socks Server forwards the request to the destination Web Server (103). The flow in the reverse direction (HTTP response) again goes via the Socks Server (104) to the Web Browser (101). In this way the Socks Server can limit the traffic to authorised transactions according to its configuration (based on some

defined security and access control policy). The Socks Server hence protects the network where Web Browser is located.

Physical View

Figure 2 is a physical view of the set-up logically described in Figure 1. In this particular example, the Web Browser (201) runs on a system (workstation) connected to an Intranet (202) network. The Intranet network comprises network devices such as IP Routers (206). The Socks Servers (203) protecting the Intranet connect both the (private) Intranet (202) and the (public) Internet (204). the destination Web Server (205) is also connected to the Internet. It is important to note that Socks Servers attach two networks and hence act as intermediaries for communications between said two networks. Multiple Socks Servers are often used in order to provide access robustness and load sharing.

IP DATAGRAM

The transfer unit of a data packet in TCP/IP is called an IP Datagram. It is made up of a header containing information for IP protocol and data that is only relevant to the higher level protocol. Figure 3 shows the format of a IP Datagram, in the environment described in Figures 1 and 2:

- (301) ***IP Datagram***. an IP Datagram is a message exchanged between 2 computer systems across a TCP/IP network. An IP Datagram is divided in 2 parts:
  - a Header, and
  - Data.

- (302) ***IP Datagram Header***. the header comprises fields such as:

  - the Type Of Service (TOS) field (310). The Type Of Service (TOS) field (310) is an indication of the quality of service requested for the IP Datagram. It can be used to provide the nature and the priority of

the IP Datagram. It can be set either by the system initiating the connection (the origin), or on the fly by a network device (for instance an IP Router) within the network.

- the Source IP Address (312) (the IP address of the computer which sends the IP Datagram).
- the Destination IP Address (the IP address of the computer which is the destination of the IP Datagram).
- The Header Checksum (311) is used by the destination of the IP datagram for checking that all fields in the IP Datagram Header have not been altered in the network. The Header Checksum is a combination of all fields in the IP Datagram Header.

The IP Datagram Header is mainly used to route the IP Datagram to its final destination.

- (303) ***IP Datagram Data***. This field comprises the data sent by the originator to the destination computer system. The destination computer system processes this data. Since the TCP/IP protocol suite is organised in layers, the IP Datagram field comprises the message relevant to the higher level protocol (which is TCP in the environment related to the invention).

- (304) ***TCP Segment***. A TCP message is usually called TCP Segment.

- (305) ***TCP Header***. A TCP Header comprises fields such as the Source Port (314) and the Destination Port (Dest Port) (315) which identify the application protocol (e.g. HTTP, FTP, Telnet, Socks) transported by TCP. This field is mainly used by the destination of the IP Datagram to determine which application must process the data transported by TCP.

- (306) ***TCP Data***. The TCP Data field comprises application data which are sent by the originator to

the destination computer system. The destination computer system processes the data. Since the TCP/IP protocol suite is organised in layers, the TCP Data part contains the information relevant to the higher level protocol which is the Application level protocol (such as HTTP, FTP, Telnet, Socks).

- (307) ***Application Level Message***. The TCP Data part of the IP Datagram contains an Application Level Message. This is for example a Socks message (for instance a "CONNECT" or a "BIND" message), a HTTP message, a FTP message, or a Telnet message. Depending on the Application level protocol, this Application Level Message can also be split in 2 parts.

- (308) ***Application Level Header***. The Application Level Header is the header relevant to the application protocol such as HTTP, FTP, Telnet.

- (309) ***Application Level Data***. This is the data part which is processed by the application responsible of handling the Application Level protocol. This is usually the data which is directly relevant to the end user (for instance, data entered by an end user on his workstation).

SOCKS CLIENT AND SOCKS SERVER

A Socks is a networking proxy protocol that allows client workstations to gain full access to hosts outside their local network while providing a high degree of security. Figure 4 shows an end user workstation (401) connected to an Intranet (402). The Socks Servers (403) that protect the Intranet attach both the (private) Intranet (402) and the (public) Internet (404). The destination Web System (405) also connects the Internet (the Web System is for instance a WEB Browser, a

FTP Server, or any system attached to the Internet that can be accessed from the Intranet).

The end user workstation (401) comprises a software program called Socks Client (406) for a secure access to the World-Wide-Web (WWW) via a Socks Server. The Socks Client creates some form of *"virtual tunnel"* between workstation and Socks Server. The Socks protocol is independent of the Application Level protocol, and can therefore be used for instance for HTTP, FTP, or Telnet communications.

When multiple Socks Servers (403) are used to access WEB Systems on the Internet, a Dispatcher System (411) is often used to balance the traffic load across the multiple Socks Servers. End user workstations (401) are then accessing the plurality of Socks Servers (403) by means of this single Dispatcher System which acts as one virtual single Socks Server.

The IP Router (410) within the Intranet routes each IP Datagram it receives towards its destination. The IP Router determines the next hop (413), using the Destination IP Address field in the IP Datagram Header.

WEB applications running on an end user workstation (such as a WEB Browser (407), an FTP Client (408), or a Telnet Client (409)) use the services of a Socks Client (406) to gain access to an outside WEB System (405). The Socks Client located on the end user workstation sends (412) each IP Datagram on the Intranet network to a destination Socks Server. IP Routers within the Intranet receive the IP Datagrams and route them (413) towards their destination. The Dispatcher System (411) which is viewed by the Intranet as a Socks server receives the IP Datagrams. The Dispatcher System selects for each IP Datagram the "best" Socks Server, and forwards each IP Datagram to the selected Socks Server. The Socks Server authenticates the user, authorizes the request, establishes a connection (415) with the outside WEB System, and then transparently forwards application data between the end user workstation and the WEB System (416).

AUTO TOS SYSTEM

The present invention relates to a system and method for setting the Type Of Service (TOS) field of IP Datagrams transporting Socks data according to the Application Level protocol used by said Socks data. Figure 5 is a general view of an end user workstation retrieving from an Auto TOS System the Type Of Service (TOS) values associated with the Application Level protocols used by Socks data according to the present invention.

An end user workstation (501) comprising a Socks Client is connected to the Intranet (502), said Intranet comprising multiple Socks Servers (503) for accessing the WEB System (505) connected to the Internet (504). An AutoTOS system (506) located within the Intranet (502) is in charge of providing the information (507) required for configuring end user workstations connected to the Intranet. Said information is comprised in a table called "TOS Definition table" (507). This table associates each Application Level protocol used by Socks data with a Type Of Service (TOS) value. For instance, the TOS Definition table may associate the FTP protocol with a TOS value of 0, the Telnet protocol with a TOS value of 1, and the HTTP protocol with a TOS value of 7. In a preferred embodiment, the TOS Definition table is a flat file and is created and maintained for instance by a Network Administrator. Before accessing (510) the World Wide Web, the end user workstation (501) gets connected (508) to the AutoTOS system and retrieves (509) the TOS Definition table located in said AutoTOS system using for instance the HTTP protocol or any file transfer protocol such as FTP. The TOS Definition table can be:

- statically defined on the AutoTOS system. In this case, the same TOS Definition table is retrieved by all end user workstations. This way, all end user workstations will use the same TOS values for the same Application Level protocols, therefore providing a consistent TOS definition within the whole Intranet. For instance, all

end user workstations within the network will use a TOS value of 0 for the FTP protocol.

- or dynamically built when the end user workstation requests the AutoTOS system to send back the TOS Definition table. The table can be for instance dynamically built by a CGI (Common Gateway Interface) computer program running on the AutoTOS system. The TOS Definition tables returned to the different end user workstations may be different according to some criteria such as the IP address of the end user workstation. For instance, one group of users (group0) may use a TOS value of 0 for FTP, while another group of users (group1) may use a TOS value of 1 for that same FTP protocol. IP Datagrams having a TOS value of 1 will be processed with a higher priority than IP Datagrams having a TOS value of 0. The FTP service provided to group1 will offer better performances than the FTP service provided to group0.

IP DATAGRAM ROUTING AND DISPATHCHING

Figure 6 is a view of the routing and dispatching of IP Datagrams according to the present invention. The end user workstation (601) comprises:

- a Socks Client (606), and
- a TCP/IP Stack (607) (the TCP/IP Stack is in charge of processing all IP Datagrams on the workstation).

This end user workstation is connected to the Intranet (602), said Intranet comprising multiple Socks Servers (603) to access the WEB System (605) connected to the Internet (604).

A Dispatcher System (615) located within the Intranet is used to dispatch IP Datagrams transporting Socks data on Socks Servers (603). The Dispatcher System which is viewed by the Intranet network as a Sock Server, receives the IP Datagrams

originated by the end user workstation (601). Then, it selects the "best" Socks Server where to forward the IP Datagrams, using information comprised in the Type Of Service (TOS) field in the IP Datagram Header. For instance an IP Datagram with a TOS value of 0 will be forwarded to a low capacity Socks Server, while an IP Datagram with a TOS value of 7 will be forwarded to another more powerful Socks Server. The IP Router (614) receives (616) IP Datagrams from the end user workstation and routes then within the Intranet towards their destination.

SOCKS TOS CLIENT

The end user workstation comprises:

- (606) a ***Socks Client***.
- (607) a ***TCP/IP Stack***. The TCP/IP Stack processes all IP Datagrams, and is in particular in charge of sending all IP Datagrams to the Intranet.
- (608) a ***Socks TOS Client*** component according to the present invention.

The Socks TOS Client determines for each IP Datagram a TOS value depending on the Application Level protocol used by the socks data transported in the IP Datagram and writte this TOS value in the Type Of Service (TOS) field of the IP Datagram. The Socks TOS Client is configured with a TOS Definition table associating with each Application Level Protocol, a particular TOS value.

When the end user workstation must send an IP Datagram comprising Socks data to a Socks Server (603), the TCP/IP Stack (607) first forwards said IP Datagram to the Socks TOS Client (608) within the workstation. The IP Datagram is then forwarded in sequence to a plurality of other components within said Socks TOS Client to perform the method according to the present invention:

- (609) a ***TOS Definition Builder*** component builds the TOS Definition table (610) that is used by the Socks TOS Client to determine the TOS value of each IP Datagram comprising socks data. The TOS Definition table is preferably retrieved from the AutoTOS system described in Figure 5 by this TOS Definition Builder component.

- (612) a ***Socks Traffic Analyser*** component analyses the IP Datagram and builds a Socks Connection table (611) with the characteristics of the Socks connections using the Application Level protocol of the IP Datagram jointly with information comprised in the TOS Definition table (610).

- (613) a ***Socks TOS Finder*** component determines the Type Of Service (TOS) field of the IP Datagram, based on the Socks Connection table (611) jointly with the Source IP Address, Source Port, Destination IP Address and Destination Port fields of the IP Datagram.

- (614) a ***Socks TOS Builder*** component writes the value determined by the Socks TOS Finder component in the TOS field of the IP Datagram.

The Socks TOS Client finally returns the IP Datagram to the TCP/IP Stack (607) component within the end user workstation. This TCP/IP component then sends (616) the IP Datagram towards its destination. The invention does not depends on the TCP/IP Stack component and does not rely on the way the IP Datagram is handled and forwarded by this TCP/IP Stack component. The IP Router (614) within the Intranet receives the IP Datagram and routes it (617) towards its destination. Possibly, the IP Router can process the IP Datagram according to the TOS field, using any existing mechanism (for instance IP Datagrams with a TOS of 7 can be routed faster than IP Datagrams with a TOS of 0). The Dispatcher System (615) which receives the IP Datagram then selects the "best" Socks Server where to forward the IP

Datagram using for instance the Type Of Service (TOS) field in the IP Datagram Header, and then forwards (518) the IP Datagram to said "best" Socks Server.

Optionally, the Socks TOS Client can be extended (for instance by means of a configuration parameter of the Socks TOS Client) to determine the TOS value and write this TOS value in the TOS field for all IP Datagrams. This way, not only IP Datagrams comprising Socks data can be processed by the Socks TOS Client. The Socks TOS Client then becomes a general "TOS Client" which is no longer specific to Socks traffic. In this case, the Application Level protocols comprised in the TOS Definition table are not interpreted by the Socks TOS Client as protocols used by socks data (for instance HTTP transported by socks) but as protocols directly used by TCP data (for instance HTTP transported by TCP).

SOCKS TOS CLIENT INTERNAL TABLES

Figure 7 depicts the different tables located in the end user workstation. These tables are used by the various components of the Socks TOS Client for analysing Socks data and for determining the value of the Type Of Service (TOS) field. The Socks TOS Client on the end user workstation uses a first table called TOS Definition table (701) created:

- either, on an AutoTOS System (506) (for instance by a Network Administrator). The table is then downloaded from the AutoTOS System to the end user workstation.
- or, directly on the end user workstation (for instance manually), before starting the Socks TOS Client.

- (701) ***TOS Definition table***. This table comprises for each Application Level Protocol, the value that must be written in the Type Of Service (TOS) field of the IP Datagram.

The Socks TOS Client uses for internal purpose a second table which is dynamically built:

- (705) ***Socks Connection table***. This table comprises the originator and the Type Of Service (TOS) of each Socks connection.

These two internal table are detailed in Figure 7.

TOS DEFINITION TABLE

The TOS Definition table (701) (a flat file in a preferred embodiment) is created by the Network Administrator in charge of the Intranet. This table associates each Application Level Protocol with a Type Of Service value. The table contains a list of records (702), each record comprising the following information:

- (703) ***Application_Level_Protocol*** (also referred to as ALP). There is one value for each Application Level protocol. Typically, a record is defined for each of the main WWW protocols including Gopher, HTTP, FTP, Telnet, SSL (this is Secure HTTP).

- (704) ***Type_Of_Service*** (also referred to as TOS). This is the value of the TOS associated with the Application Level Protocol (703). This TOS value will be written in the TOS field of the IP Datagrams transporting the Application Level Protocol (703) over Socks. For instance, FTP can be associated with a TOS value of 0, while HTTP can be associated with a TOS value of 7. When the TOS field is then used within the Intranet to differentiate the Socks traffic, the HTTP traffic will be treated with a higher priority than the FTP traffic.

The TOS Definition table also comprises default values (Type_Of_Service default values) for ALP value is not explicitly defined in the table.

SOCKS CONNECTION TABLE

The Socks Connection table (705) is an internal table built by the Socks TOS Client. It is used to store the originator and the Type Of Service of each Socks connection. The table contains a list of records (706), each record providing the following information:

- (707) ***Cx_Source_IP_Address***. This is the IP address of the system which is the originator of the Socks connection (this system is called *"originator"* or *"origin device"*).

- (708) ***Cx_Source_Port***. This is the number of the Port identifying the program (the application) running on the origin device and which is the originator of the Socks connection

- (709) ***Cx_Dest_IP_Address***. This is the IP address of the system which is the destination of the Socks connection (this system is called *"destination"* or *"destination device"*).

- (710) ***Cx_Dest_Port***. This is the number of the Port identifying the destination program (the application) running on the destination device.

  The combination of Cx_Source_IP_Address, Cx_Source_Port, Cx_Dest_IP_Address, and Cx_Dest_Port identifies one and only one socks connection.

- (711) ***Cx_TOS***. This is the value of the TOS field for the Socks connection uniquely identified by Cx_Source_IP_Address (707), Cx_Source_Port (708), Cx_Dest_IP Address (709) and Cx_Dest_Port (710).

The combination of Cx_Source_IP_Address and Cx_Source_Port identifies the source application running on the source device

at the origin of the socks connection. The combination of Cx_Dest_IP_Address and Cx_Dest_Port identifies the destination application running on the destination device at the destination of the socks connection. To identify a TCP connection (not only socks), the combination of Cx_Source_IP_Address, Cx_Source_Port, Cx_Dest_IP_Address, and Cx_Dest_Port must be used.

TOS DEFINITION BUILDER

The TOS Definition Builder component of the Socks TOS Client is preferably a computer program running on the end user workstation. This component is in charge of building the Socks Definition table (601) which associates each Application Level protocol with a TOS value. The TOS Definition Builder is the first component initialized when the Socks TOS Client is started on the end user workstation. Figure 8 is a flow chart which refers to the internal logic of the TOS Definition Builder component. This component:

- (801) opens a connection, preferably a TCP connection, with the AutoTOS system (506). The AutoTOS system is identified by means of a configuration parameter stored in the Socks TOS Client (preferably the IP Address of the AutoTOS system).

- (802) retrieves the TOS Definition table (803) from the AutoTOS system. Preferably, the protocol used to retrieve the table is FTP or HTTP.

- (804) stores the TOS Definition table locally on the workstation (501).

- (805) closes the connection with the AutoTOS system.

Optionally, the TOS Definition table can be manually placed on end user workstation instead of being retrieved from an AutoTOS system.

Optionally, the TOS Definition table is periodically updated. The TOS Definition Builder component can periodically retrieve the TOS Definition table from the AutoTOS system. An update of the TOS Definition table can also be automatically sent to end user workstations by the AutoTOS system.

SOCKS TRAFFIC ANALYSER

The Socks Traffic Analyser component of the Socks TOS Client is preferably a computer program running on the end user workstation. This component is in charge of:

- determining the Application Level protocol of each IP Datagram received by the Socks TOS Client, and
- building the Socks Connection table (705).

Figure 9 is a flow chart which refers to the internal logic of the Socks Traffic Analyser component. This component:

- (901) retrieves an IP Datagram from the TCP/IP Stack component.

- (902) tests whether the IP Datagram transports Socks traffic or not. The test preferably uses the Destination Port field of the TCP Header comprised in the Data part of the IP Datagram. The Destination Port is compared to the Port that uses the Socks protocol (by default, the Port used by Socks is 1080). For instance, the Port used by the Socks protocol can be a configuration parameter of the Socks TOS Client. If the Destination Port in the TCP Header is equal to the Port used by Socks, then the IP Datagram transports Socks traffic, otherwise the IP Datagram does not transports Socks traffic and does not need to be processed by the Socks TOS Client.

  - If the IP Datagram does not transport Socks traffic:

- (903) the Socks TOS Client is not involved in the processing of non Socks traffic. The IP Datagram is directly returned to the TCP/IP Stack, which then sends it towards its destination.
  The processing by the Socks TOS Client of the IP Datagram is completed. The Socks TOS Client waits for the next IP Datagram.

- If the IP Datagram transports Socks traffic:

  - (904) identifies the socks connection. The originator and the destination of the IP Datagram can be identified directly or by the information comprised in the IP Header and TCP Header:

    - Datagram_Source_IP_Address = Source IP Address field (in IP Header).
    - Datagram_Source_Port = Source Port field (in TCP Header).
    - Datagram_Dest_IP_Address = Dest IP Address field (in IP Header).
    - Datagram_Dest_Port = Dest Port field (in TCP Header).

  - (905) checks if the IP Datagram is a Socks CONNECT message. A Socks CONNECT message is identified by the CD field in the Application Level message (CD = 1).

    - If the IP Datagram is a not a Socks CONNECT message:

      The IP Datagram therefore belongs to a Socks connection which is already established. This means that a CONNECT message has already been received and the corresponding record has been created in the Socks Connection table.

      - (906) calls the Socks TOS Finder component.

- If the IP Datagram is a Socks CONNECT message:

  - (907) determines from the IP Datagram (in the Destport field of the Socks CONNECT message), the Application Level protocol (ALP) transported by the Socks connection:

    - Datagram_ALP = Destport field (in Socks CONNECT message)

  - (908) retrieves all records from the TOS Definition table (909). The table is preferably read only once for all and cached by the Socks Traffic Analyser at configuration time, in order to minimize the impact on performances.

  - (910) finds the record corresponding to the Application Level protocol (ALP) of the IP Datagram. This is the record with:

    - Application_Level_Protocol = Datagram_ALP

  - (911) determines from that record the TOS of the IP Datagram:

    - Datagram_TOS = Type_Of_Service

  - (912) saves in a new record within the Socks Connection table (913) the information for the Socks connection:

    - Cx_Source_IP_Address = Datagram_Source_IP_Address
    - Cx_Source_Port = Datagram_Source_Port
    - Cx_Dest_IP_Address = Datagram_Dest_IP_Address
    - Cx_Dest_Port = Datagram_Dest_Port

- Cx_TOS = Datagram_TOS

- (906) calls the Socks TOS Finder component

Optionally, all IP Datagrams not comprising Socks data can also be processed by the Socks Traffic Analyser. In this case, the Datagram_ALP is identified by the Destination Port field in the TCP Header.

SOCKS TOS FINDER

The Socks TOS Finder component of the Socks TOS Client is preferably a computer program running on the end user workstation. This component determines a TOS value for each IP Datagram, using the Socks Connection table (705). Figure 10 is a flow chart which refers to the internal logic of the Socks TOS Finder component. This component:

- (1001) retrieves the IP Datagram forwarded by the Socks Traffic Analyser component, along with the Datagram_Source_IP_Address, the Datagram_Source_Port, the Datagram_Dest_IP_Address, and the Datagram_Dest_Port.

- (1002) retrieves all records from the Socks Connection table (1003).

- (1004) finds the record which corresponds to the Socks connection. This record is identified by (four conditions):

  - Cx_Source_IP_Address = Datagram_Source_IP_Address
  - Cx_Source_Port = Datagram_Source_Port
  - Cx_Dest_IP_Address = Datagram_Dest_IP_Address
  - Cx_Dest_Port = Datagram_Dest_Port

- (1005) retrieves the Type Of Service (TOS) which corresponds to that Socks connection, from that record.

This is the value of the TOS field that must be written in the IP Datagram:

- Datagram_TOS = Cx_TOS

- (1006) removes from the Socks Connection table (1003) records of closed Socks connections (terminated connections). Closed Socks connections are detected for instance using the FIN and ACK indications in the TCP Header. Optionally, a Socks connection is considered closed after a certain period of time without IP Datagram on that Socks connection (this timer value can be for instance a configuration parameter of the Socks TOS Client). Any other existing algorithm to detect closed or half closed (for instance when one extremity of the connection has abnormally terminated) TCP connections can also be used to remove such connections from the table.

- (1007) calls the Socks TOS Builder component

SOCKS TOS BUILDER

The Socks TOS Builder component of the Socks TOS Client is preferably a computer program running on the end user workstation. This component writes in the Type Of Service (TOS) field of each IP Datagram the TOS value determined by the Socks TOS Finder component. Figure 11 is a flow chart which refers to the internal logic of the Socks TOS Builder component. This component:

- (1101) retrieves the IP Datagram forwarded by the Socks TOS Finder component, along with the Datagram_TOS.

- (1102) sets the TOS field of the IP Datagram with the Datagram_TOS. The Datagram_TOS overwrites the value of the TOS field present in the received IP Datagram. Optionally (for instance by means of a configuration

information), the TOS value written in the TOS field of the IP Datagram can be a combination of both:

- the TOS value in the IP Datagram forwarded by the TCP/IP Stack, and
- the TOS value determined by the Socks TOS Finder component.

- (1103) calculates the new value of the Header Checksum field in the IP Datagram Header. The Header Checksum is a combination of all fields in the IP Datagram Header and is used by the destination of the IP Datagram to make sure that all fields in the IP Datagram Header have not been altered in the network. Since one of the IP Datagarm Header fields (the TOS) used to calculate the Header Checksum has been modified by the Socks TOS Builder, the Header Checksum received in the IP Datagram must be updated. The new value is calculated using the existing algorithm used by any program which builds IP Datagrams.

- (1104) writes the new value of the Header Checksum field in the IP Datagram.

- (1105) forwards the IP Datagram to the TCP/IP Stack component. The TCP/IP Stack then sends the IP Datagram towards its destination. The Socks TOS Client exits and waits for the next IP Datagram.

ADVANTAGES

The present inventions provides the following advantages :

- The Type Of Service (TOS) field of IP Datagrams is representative of the Application Level protocol used for transporting socks data. Any network device, such as a Dispatcher System, can use this TOS field for differentiating IP Datagrams according to the Application

Level protocol used for transporting Socks data. For instance, HTTP traffic for the WEB can be set with a TOS value of 7, while batch FTP traffic can be set with a TOS value of 0. A Dispatcher System can then use the TOS field to dispatch:

- HTTP traffic to a high capacity Socks Server, and
- FTP traffic to a low capacity Socks Server with low capacity.

Thus, an optimized response time and availability can be provided to HTTP traffic.

- Network devices such as IP Routers may use the TOS field of the IP Datagrams to determine some IP Datagram characteristics such as the priority. Since the present invention associates a TOS value with the Application Level protocol, it therefore enables IP Routers to process IP Datagrams according to the Application Level protocol used by the Socks data they transport. For instance, IP Routers within the network may already process and route IP Datagrams with a TOS value of 7 faster than IP Datagrams with a TOS value of 0. When HTTP is associated with TOS 7 and FTP is associated with TOS 0, the IP Router will route and process faster the HTTP traffic. This mechanism provides an optimized response time for the HTTP WEB traffic.

- The definition of TOS values according to the Application Level protocols used by Socks data (the TOS Definition table) can be centrally administered on the AutoTOS system. A Network Administrator can control said TOS Definition table for all end user workstations.

- The Definition TOS table can be periodically and automatically retrieved from the AutoTOS system by the end user workstations. This way, updates of the TOS Definition table on the AutoTOS system (either manually by a Network Administrator or by automated processes) are automatically transferred to end user workstations.

For instance, in case of heavy network congestion, the TOS value for the Telnet traffic may be decreased from 5 to 0 (assuming 0 corresponds to the lowest priority for IP Datagrams which can be discarded within the network) in the TOS Definition table located on the AutoTOS system. This updated TOS Definition table is then automatically retrieved by all end user workstations (or distributed to all end user workstations). Thus, the Telnet traffic can be discarded within the network. As a consequence, the network congestion can be reduced and protocols with higher priority such as HTTP have a better chance to survive the congestion.

- The association between TOS value and Application Level protocol (TOS Definition table) can be dynamically built when the end user workstation get connected to the AutoTOS system. Different TOS Definition tables can be returned to different end user workstations depending on some criteria such as the IP address of the end user workstation. For instance, one group of end users (group0) may then use a TOS value of 0 for FTP, while another group of end users (group1) may use a TOS value of 1 for that same FTP protocol. IP Datagrams having a TOS value of 1 will be handled with a higher priority than IP Datagrams having a TOS value of 0. The FTP service provided to group1 will offer better performances than the FTP service provided to group0.

- The TOS value is written in the TOS field of IP Datagrams by end user workstations and not by any network device within the Intranet. Since companies usually own their end user workstations but may not own their intranet backbone network, it can be easier for them to implement a policy for associating TOS values and Application Level protocols on their end user workstations rather than on any network device. For instance, companies which do not own their Intranet backbone network but instead lease it from a Service Provider, can implement a policy on their end user workstations for said association. This way, they do not

have to rely on their Network Provider for providing such policy.

- The TOS field in not modified within the network by the network devices (such as IP Routers). There is therefore no impact on the performance of these network devices.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood that various changes in form and detail may be made therein without departing from the spirit, and scope of the invention.

Claims

1. A method for setting the value of a Type of Service (TOS) field in an Internet Protocol (IP) datagram (301) according to the application protocol level of socks data (303) transported in said IP datagram, said IP datagram being sent from a source application on a source device to a destination application on a destination device, said method comprising the steps of:

- identifying (1004) the source device address of the source device,
- identifying (1004) the destination device address of the destination device,
- identifying (1004) the source application address of the application on the source device,
- identifying (1004) the destination application address of the application on the destination device,
- determining (1005) a Type of Service (TOS) value referring to a first table (705), said first table comprising for each socks connection (706) identified by:
  - a source device address (707),
  - a destination device address (709),
  - a source application address (708), and
  - a destination application address (710),

  a Type Of Service (TOS) value (711).
- writing (1002) the Type Of Service (TOS) value in the Type Of Service (TOS) field (310) of the IP datagram (301).

2. The method according to the preceding claim wherein the IP datagram comprises a Source IP Address field (312) and a Destination IP Address field (313) in an IP header (302) and a Source Port field (314) and a Destination Port field (315) in a Transmission Control Protocol (TCP) header (305),

the step of identifying (1004) the source device address of the source device comprising the further step of:

- retrieving the source device address in the Source IP Address field (312) of said IP datagram,

the step of identifying (1004) the destination device address of the destination device comprising the further step of:

- retrieving the destination device address in the Destination IP Address field (313) of said IP datagram,

the step of identifying (1004) the source application address of the application on the source device comprising the further step of:

- retrieving the source application address in the Source Port field (314) of said IP datagram,

the step of identifying (1004) the destination application address of the application on the destination device comprising the further step of:

- retrieving the destination application address in the Destination Port field (315) of said IP datagram.

3. The method according to any one of the preceding claim wherein said step of determining (1005) a Type Of Service (TOS) value comprises the preliminary steps of:

- determining whether the IP datagram comprises a message for establishing a new socks connection, in particular a socks CONNECT message, or not,

---

- if the IP datagram comprises a message for establishing a new socks connection, in particular a socks CONNECT message:

  - updating the first table (705) with the new socks connection identified by:
    - the source device address,
    - the destination device address,
    - the source application address,
    - the destination application address,

    of the IP datagram,

  - identifying (907) the application level protocol from the IP datagram,

  - determining (911) a Type of Service (TOS) value referring to a second table (701), said second table comprising for each application level protocol (703) a Type Of Service (TOS) value (704),

  - associating (912) in said first table (705) said socks connection (706) with said Type Of Service (TOS) value (711).

4. The method according to any one of the preceding claims wherein said first table (705) is dynamic and comprises for each socks connection (706):

- a source device address (707) identifying the source device,

- a source application address (708) identifying the source application,

- a destination device address (709) identifying the destination device,
- a destination application address (710) identifying the destination application,
- a Type Of Service (TOS) value (711).

5. The method according to any one of the preceding claims wherein said second table (701) comprises for each application level protocol (702):

- a Type Of Service (TOS) value (704).

6. The method according to any one of the preceding claims comprising the initial steps of:

- configuring (801..805) said second table (701),
- defining a default Type Of Service (TOS) value for application level protocols not defined in said second table.

7. The method according to any one of the preceding claims wherein the step of configuring said second table (701) comprises the step of:

- retrieving (802) said second table (701) from a server system (506) within the network.

8. The method according to any one of the preceding claims wherein the step of configuring said second table (701) comprises the step of:

- retrieving (802) updates of said second table (701) from said server system (506) within the network.

9. The method according to any one of the preceding claims wherein the step of configuring said second table (701) comprises the step of:

- receiving said second table (701) from said server system (506) within the network.

10. The method according to any one of the preceding claims wherein the step of configuring said second table (701) comprises the step of:

- receiving updates of said second table (701) from said server system (506) within the network.

11. The method according to any one of the preceding claims wherein the step of configuring said second table (701) comprises the step of:

- locally storing (804) said second table (701) and/or updates of said second table within the source device.

12. The method according to any one of the preceding claims wherein the IP datagram comprises a header checksum field (311), said step of writing the Type Of Service (TOS) value in the Type Of Service field comprising the further step of:

- computing (1103) the value of the header checksum field (311) according to the Type Of Service (TOS) value.

- writing (1104) said computed value in the header checksum field (311).

13. A system, in particular a source device (601), comprising means adapted for carrying out the method according to any one of the preceding claims.

14. A computer readable medium comprising instructions for carrying out the method according to any one of claims 1 to 12.

SYSTEM AND METHOD FOR ACCESSING A SOCKS SERVER FROM AN END USER WORKSTATION IN AN IP NETWORK

Abstract

The present invention discloses a method and system in a source device, in particular an end user workstation, for setting the value of a Type of Service (TOS) field in an Internet Protocol (IP) datagram (301) according to the application protocol level of socks data (303) transported in this IP datagram. The IP datagram is sent from a source application on a source device to a destination application on a destination device. The present method comprises the steps of:

- identifying (1004) the source device,
- identifying (1004) the destination device,
- identifying (1004) the application on the source device,
- identifying (1004) the application on the destination device,
- determining (1005) a Type of Service (TOS) value referring to a first table (705), said first table comprising for each socks connection (706) identified by:
  - a source device address (707),
  - a destination device address (709),
  - a source application address (708), and
  - a destination application address (710),

  a Type Of Service (TOS) value (711).
- writing (1002) the Type Of Service (TOS) value in the Type Of Service (TOS) field (310) of the IP datagram (301).

Figure 6

Logical View of an End User Accessing the World-Wide-Web

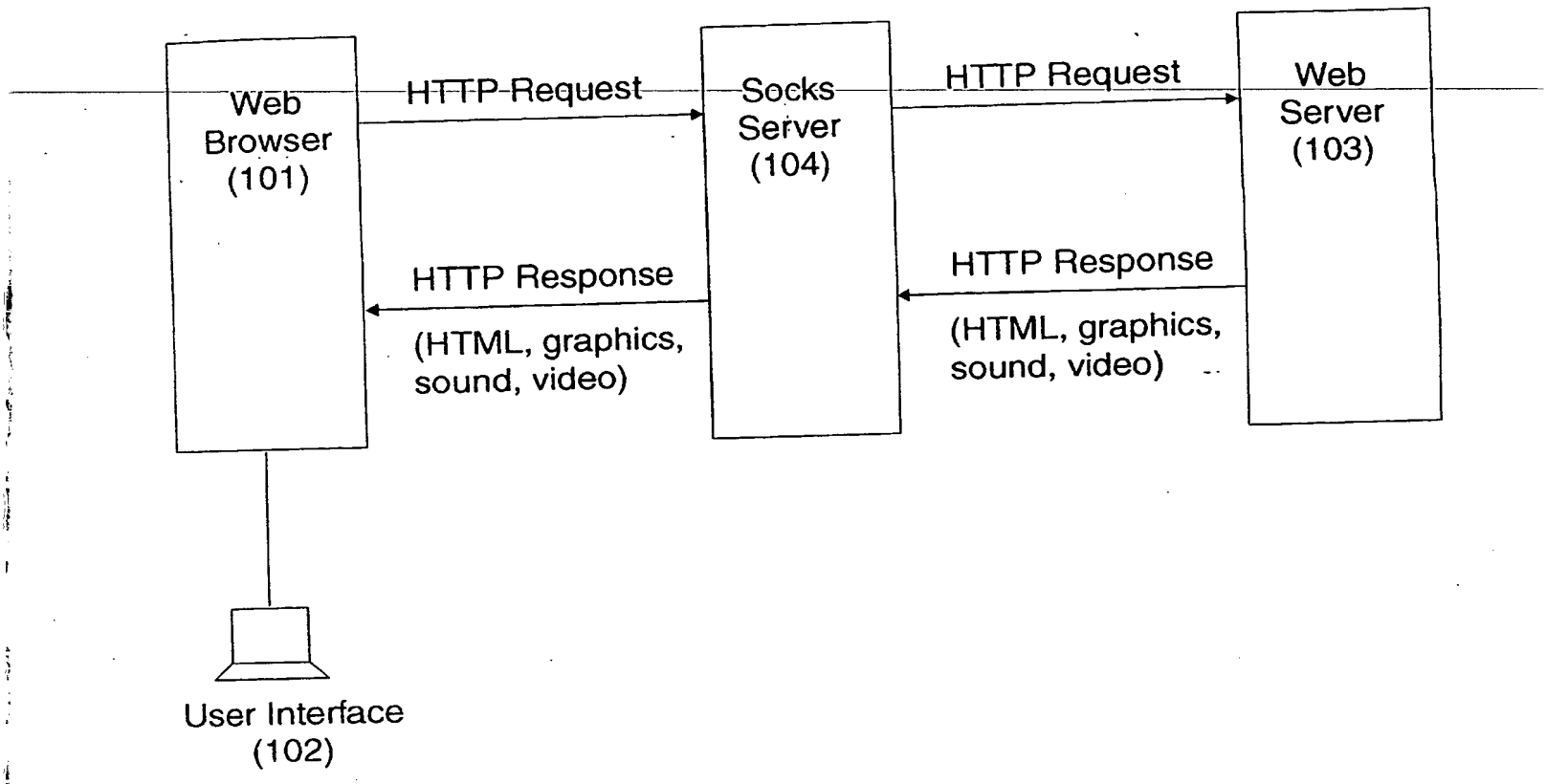

FIG. 1

General Physical View of an End User Accessing the World-Wide-Web

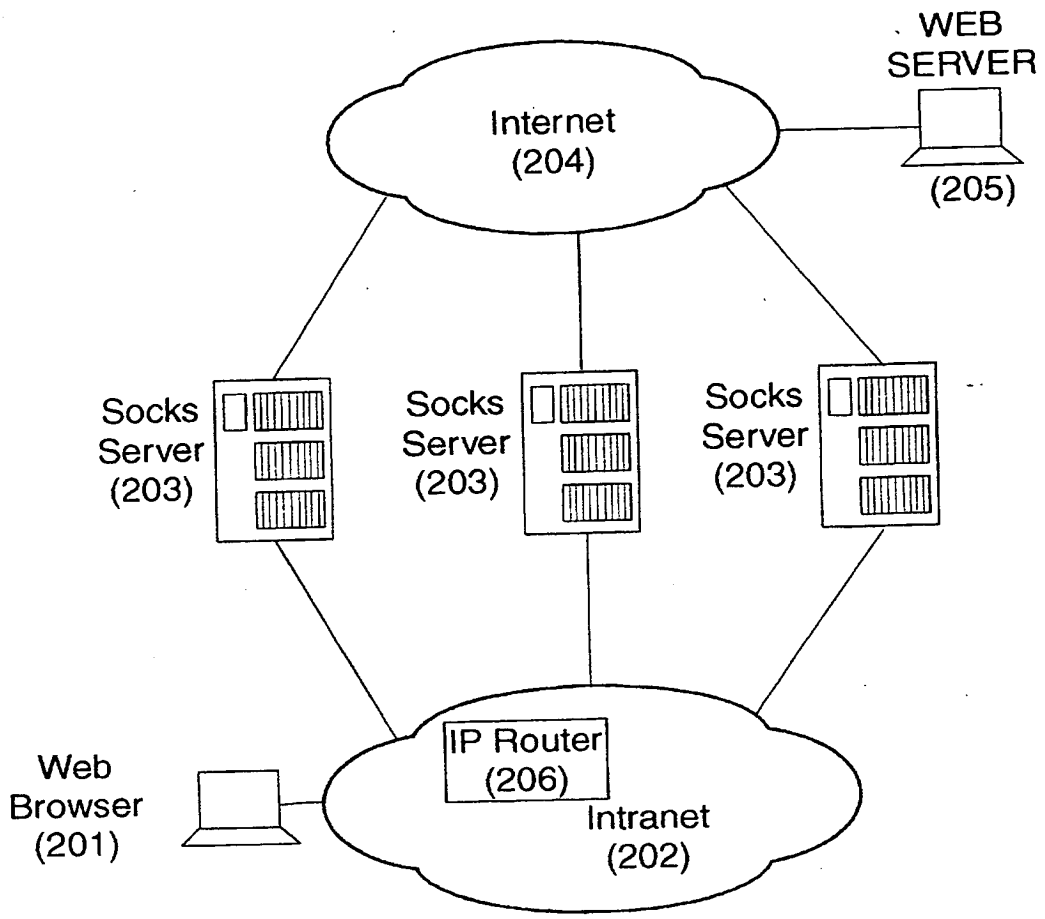

FIG. 2

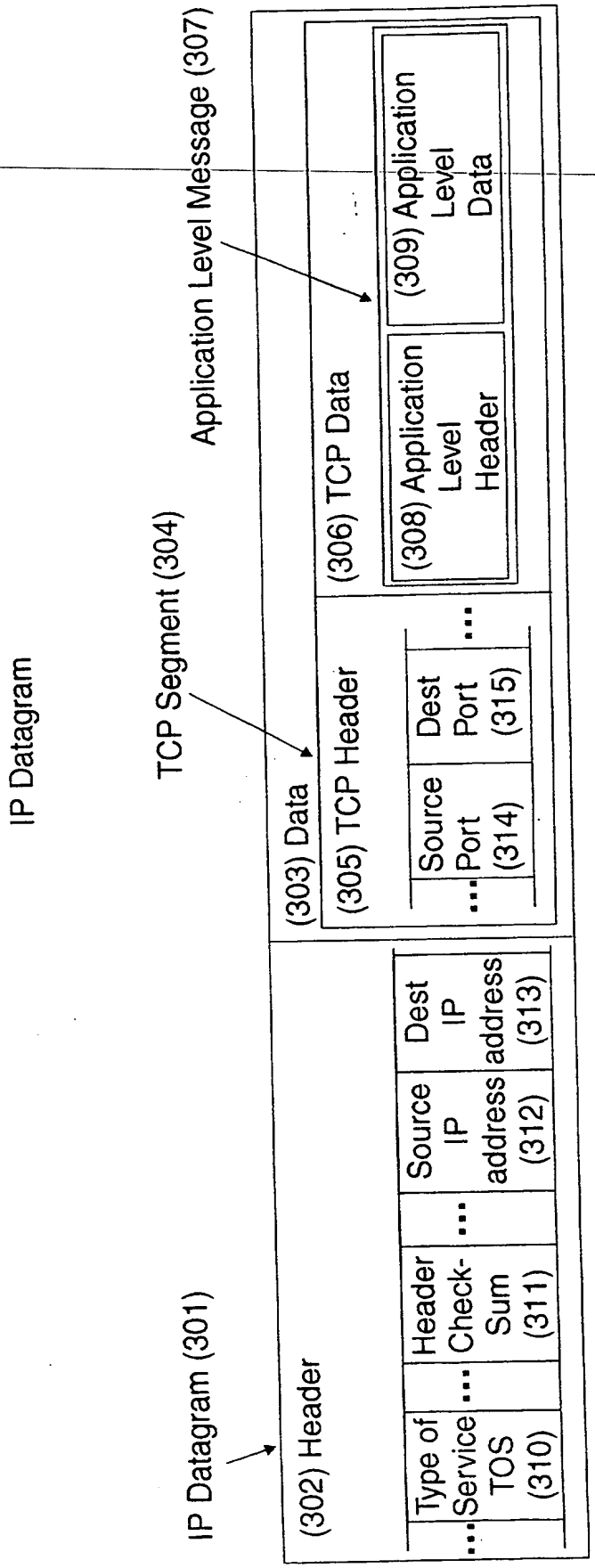
IP Datagram
IP Datagram (301)
(302) Header
Type of Service TOS (310)
...
Header Check-Sum (311)
...
Source IP address (312)
Dest IP address (313)
TCP Segment (304)
(303) Data
(305) TCP Header
Source Port (314)
Dest Port (315)
...
(306) TCP Data
Application Level Message (307)
(308) Application Level Header
(309) Application Level Data

FIG. 3

Socks Client and Socks Servers

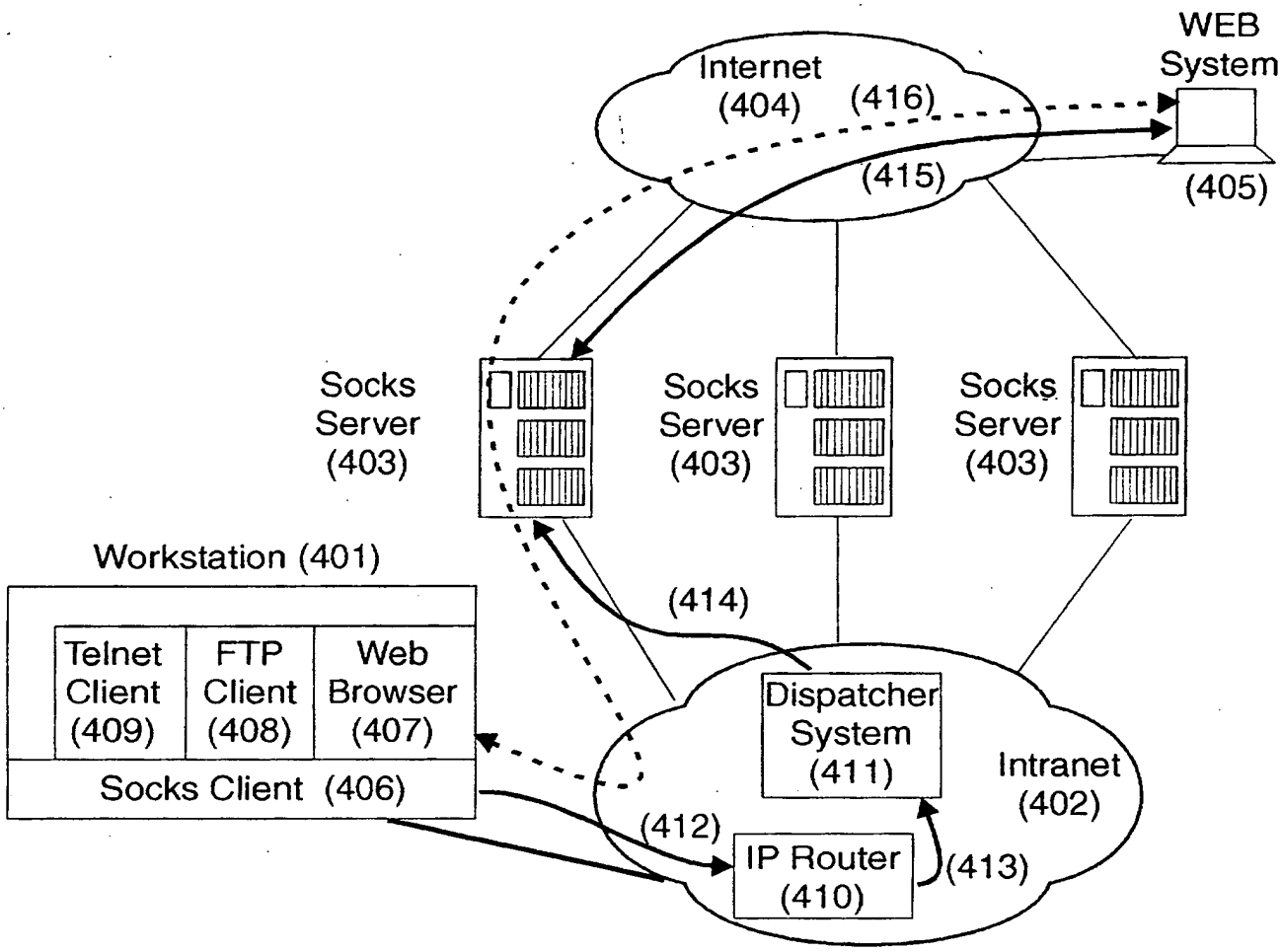

FIG. 4

End User Workstation Retrieving TOS Definition

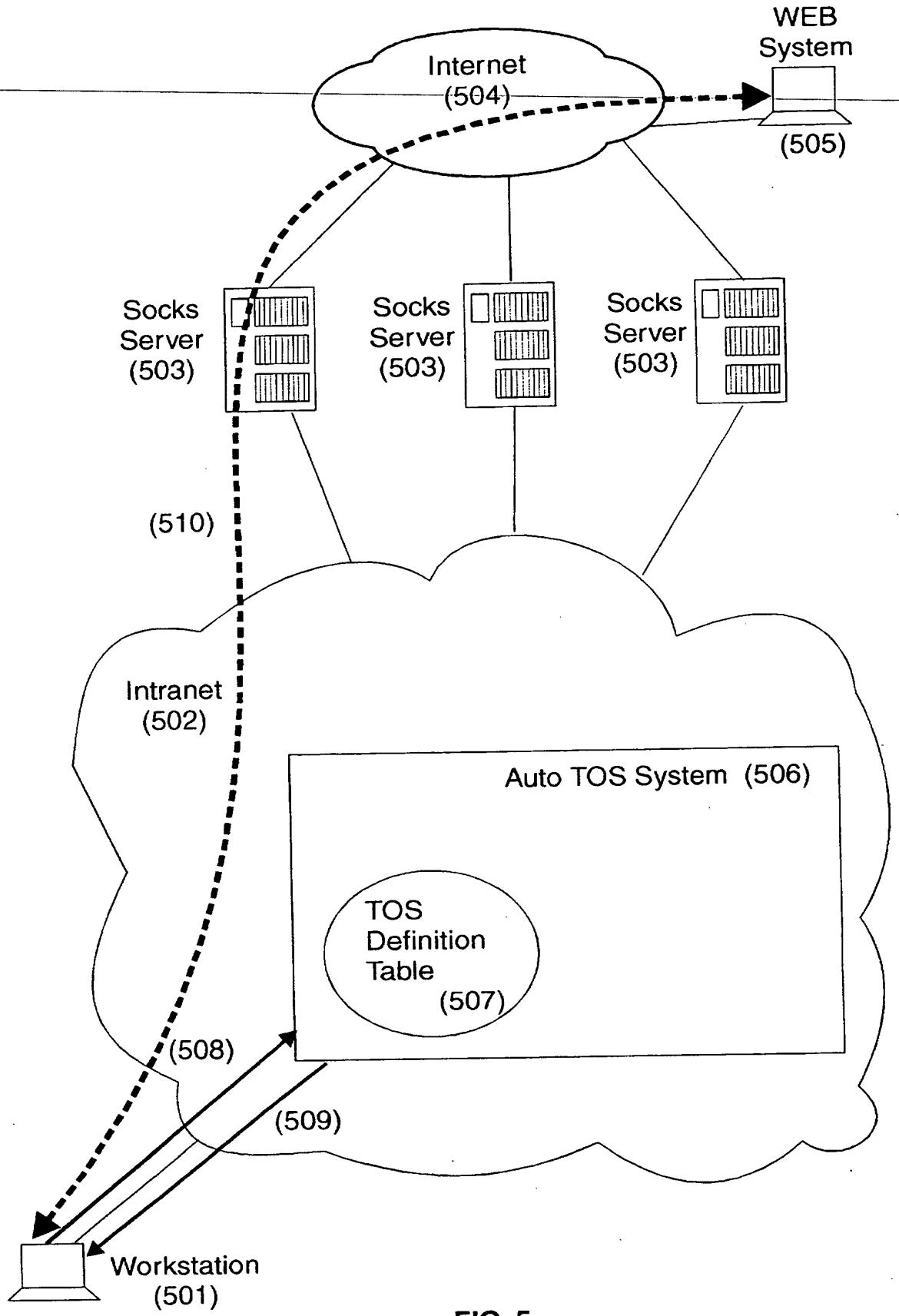

FIG. 5

End User Workstation Processing the TOS Field

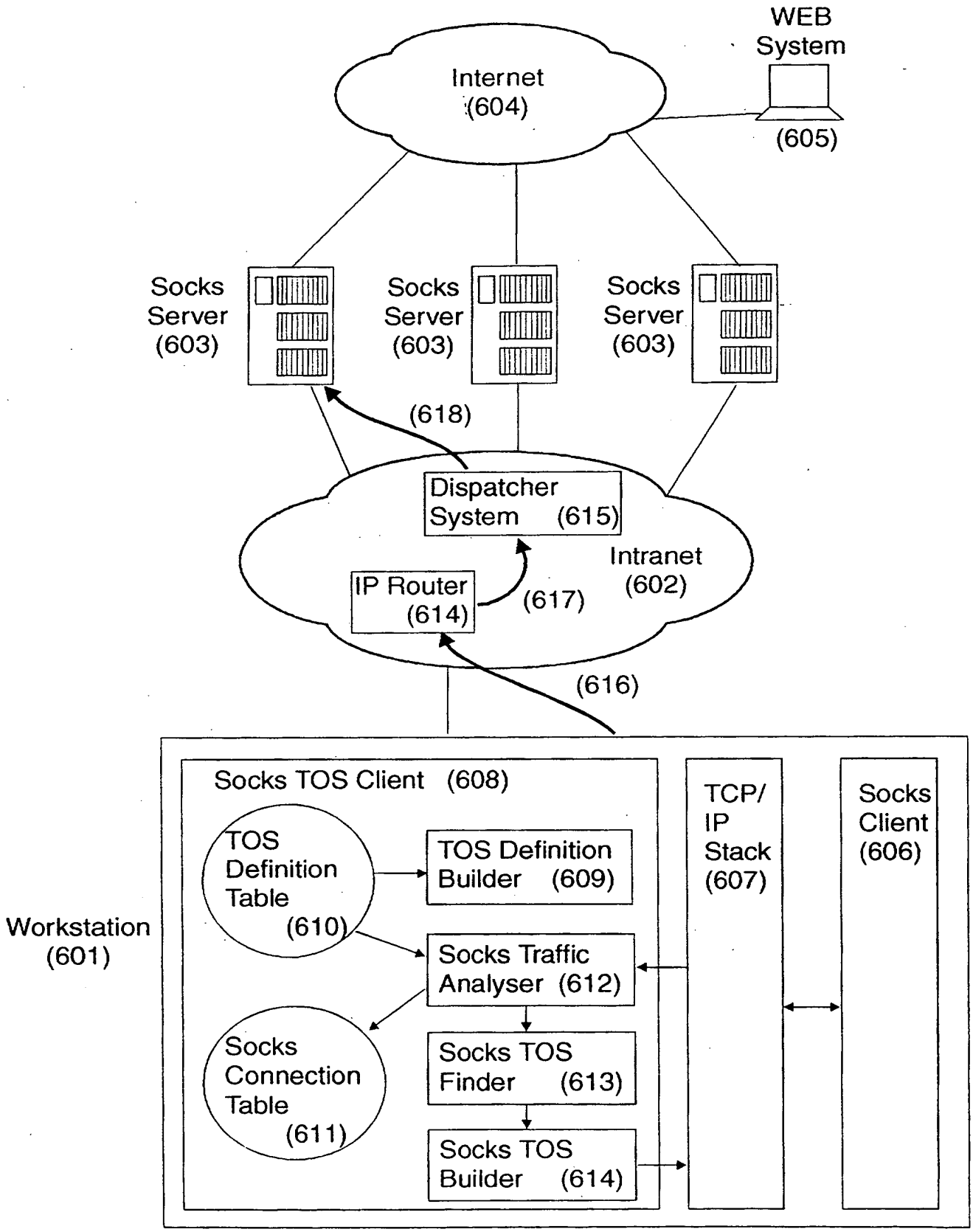

FIG. 6

Socks TOS Client Tables

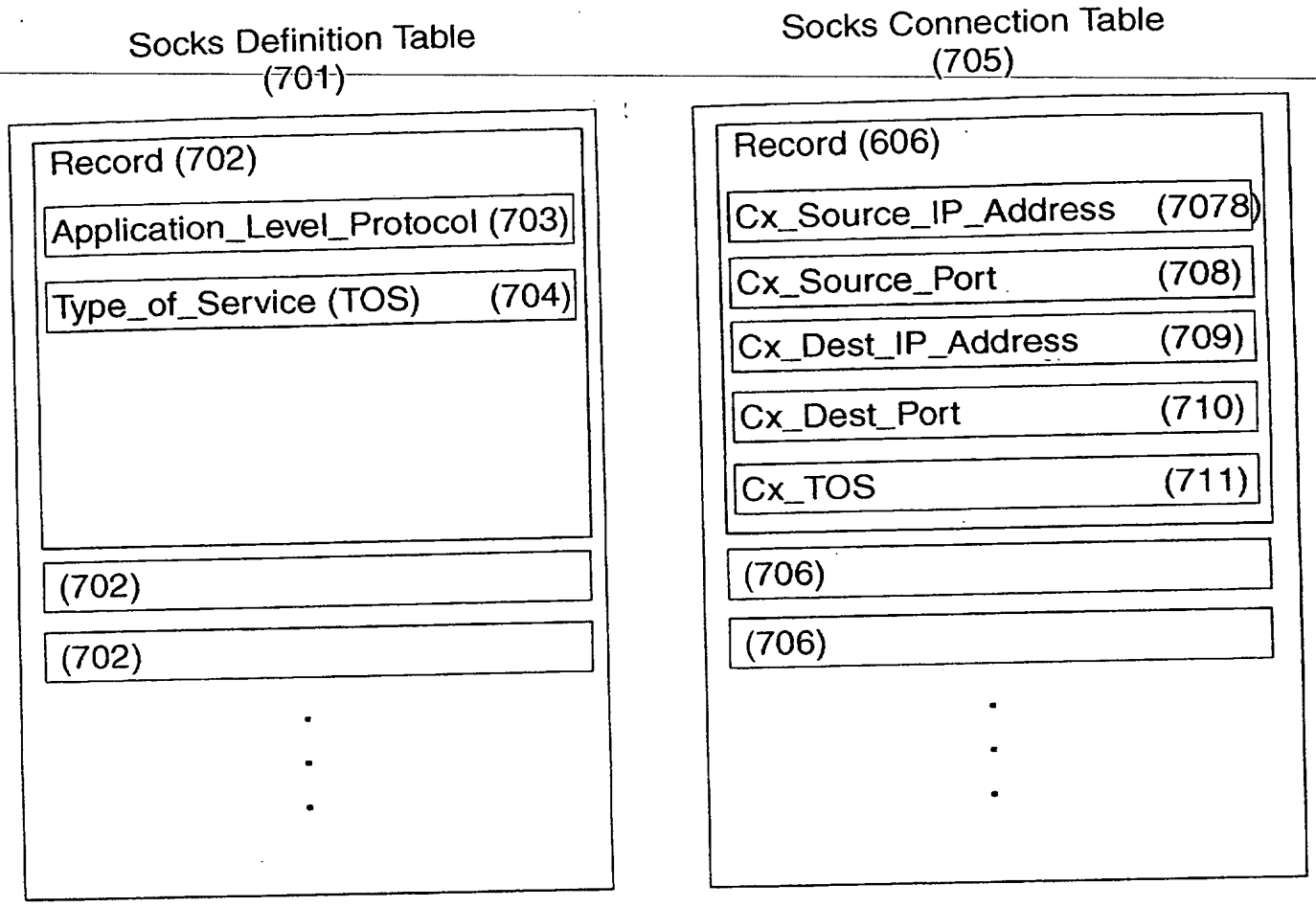

FIG. 7

Socks Definition Builder Component

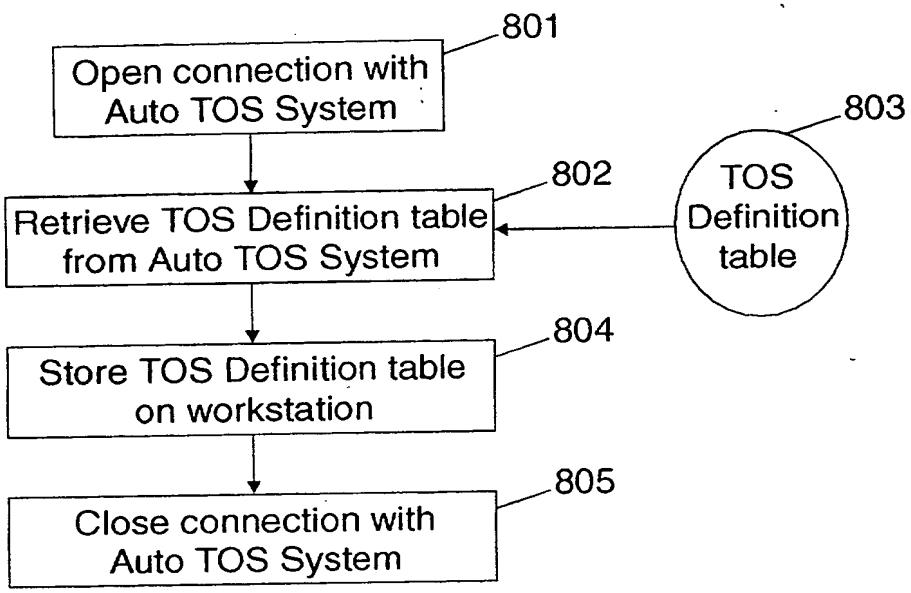

FIG. 8

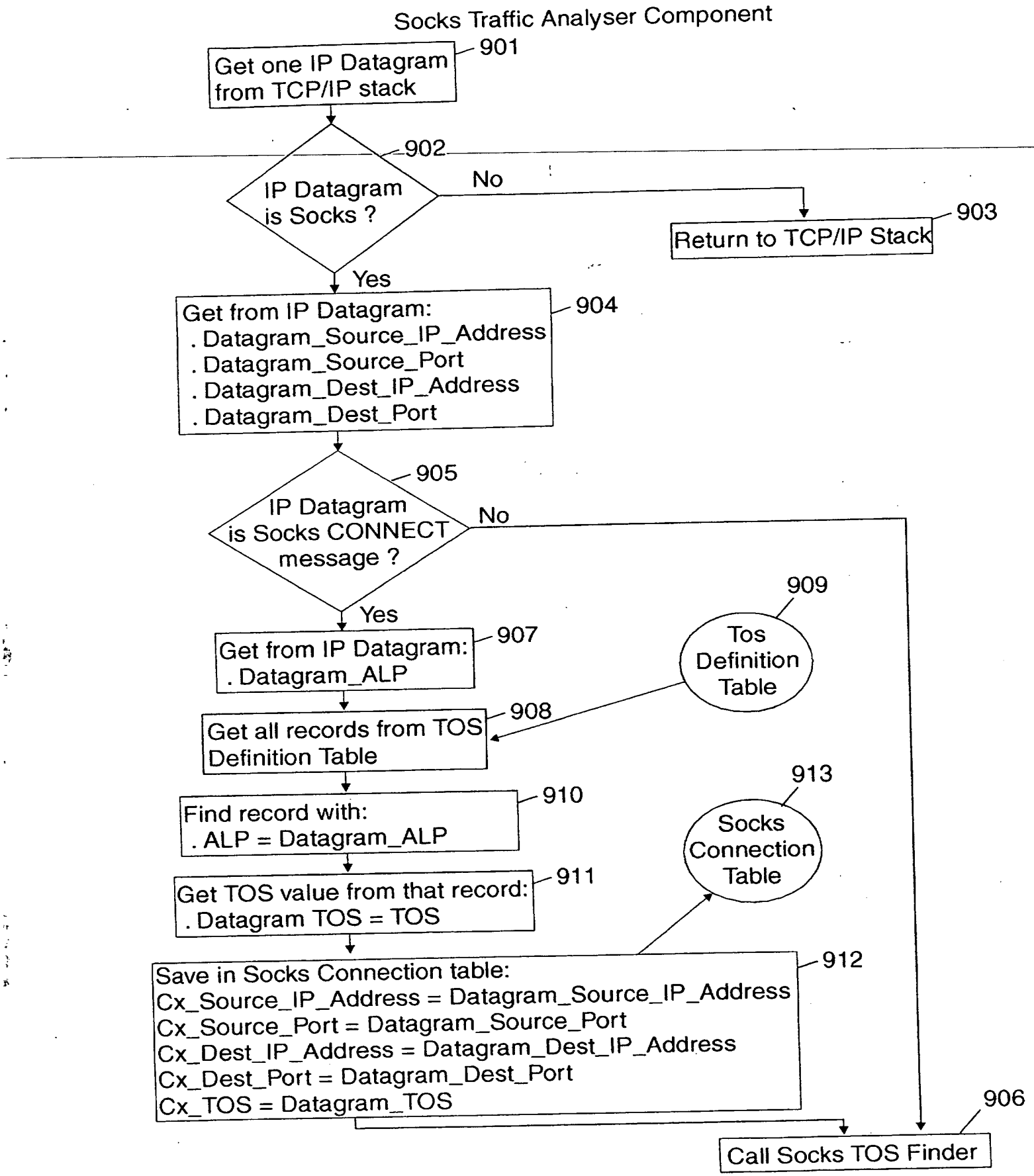

FIG. 9

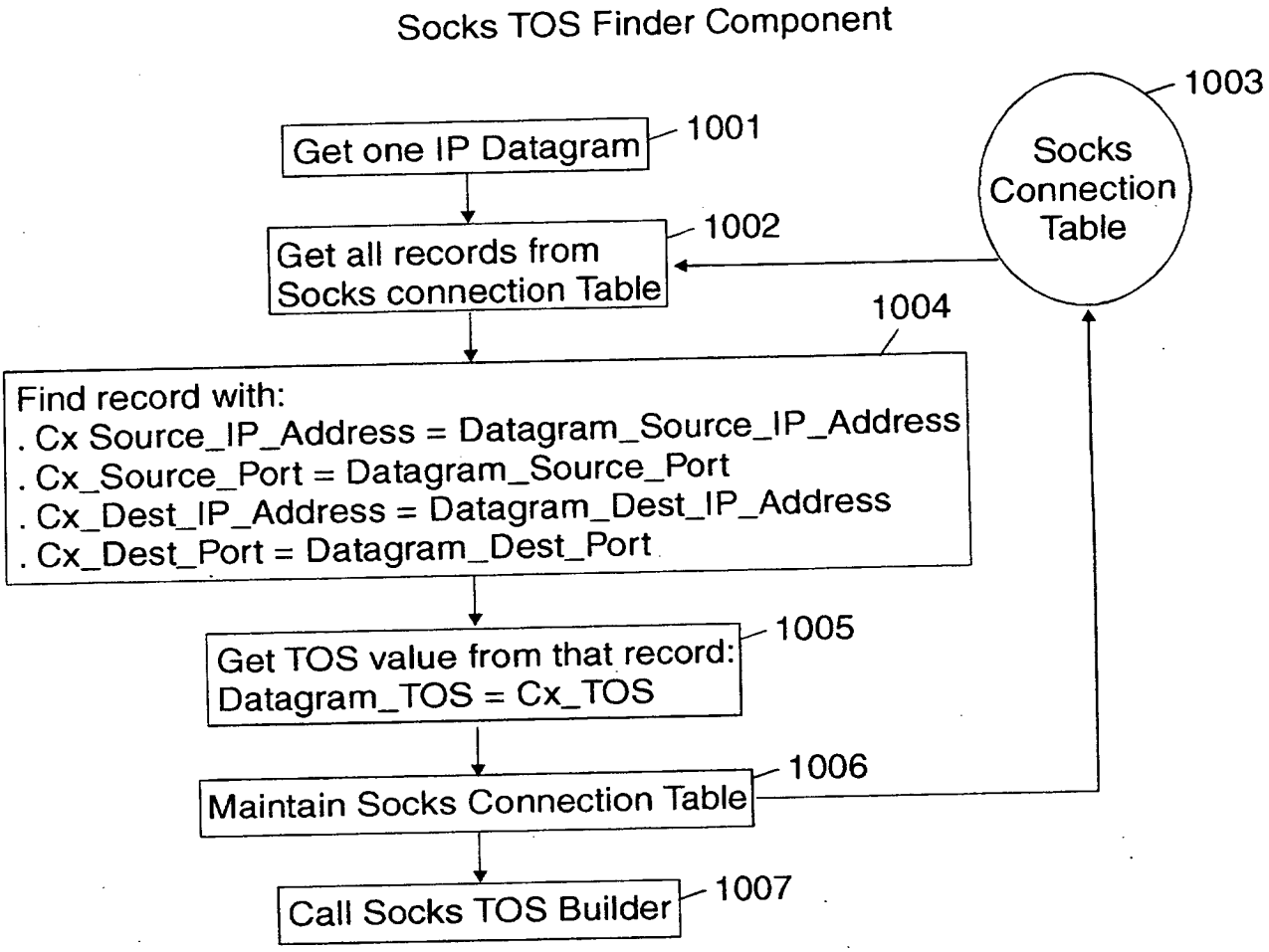
Socks TOS Finder Component
Get one IP Datagram
1001
Get all records from Socks connection Table
1002
Socks Connection Table
1003
Find record with:
. Cx Source_IP_Address = Datagram_Source_IP_Address
. Cx_Source_Port = Datagram_Source_Port
. Cx_Dest_IP_Address = Datagram_Dest_IP_Address
. Cx_Dest_Port = Datagram_Dest_Port
1004
Get TOS value from that record:
Datagram_TOS = Cx_TOS
1005
Maintain Socks Connection Table
1006
Call Socks TOS Builder
1007

FIG. 10

Socks TOS Builder Component

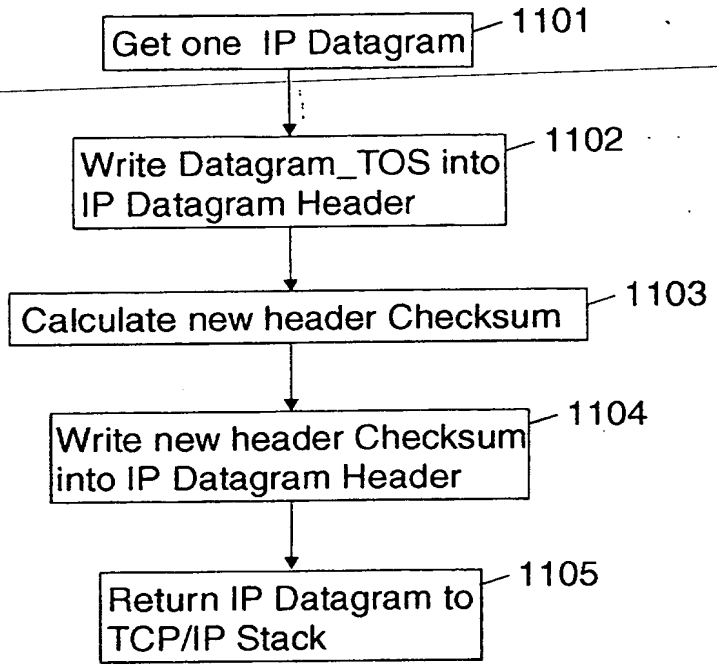

FIG. 11